Journal of Symbolic Computation 47 (2012) 1089–1108

Contents lists available at SciVerse ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

Projection and scope-determined circumscription

Christoph Wernhard

Technische Universität Dresden, Germany

ARTICLE INFO

Article history:
Received 22 February 2011
Accepted 15 July 2011
Available online 23 December 2011

Keywords:
Circumscription
Second-order quantifier elimination
Projection
Forgetting
Strongest necessary and weakest sufficient condition

ABSTRACT

We develop a semantic framework that extends first-order logic by literal projection and a novel second semantically defined operator, *raising*, which is only slightly different from literal projection and can be used to define a generalization of parallel circumscription with varied predicates in a straightforward and compact way. We call this variant of circumscription *scope-determined*, since like literal projection and raising its effects are controlled by a so-called *scope*, that is, a set of literals, as parameter. We work out formally a toolkit of propositions about projection, raising and circumscription and their interaction. It reveals some refinements of and new views on previously known properties. In particular, we apply it to show that well-foundedness with respect to circumscription can be expressed in terms of projection, and that a characterization of the consequences of circumscribed propositional formulas in terms of literal projection can be generalized to first-order logic and expressed compactly in terms of new variants of the strongest necessary and weakest sufficient condition.

1. Introduction

First-order logic provides a well-researched, quite general, and to some degree mechanizable framework for formalizing knowledge. However, for many tasks in knowledge representation it is difficult, if not impossible, to formalize the involved knowledge purely in first-order logic. One way out is to move on to special logics for knowledge representation, another way is to consider slight extensions of first-order logic that are powerful enough to express many concepts of knowledge representation. The problem addressed in the paper is to investigate some extensions of this kind, in particular variants of second-order quantification and circumscription.

The problem is primarily important because slightly extended first-order logic has the potential of being the foundation for a single knowledge representation system that provides different knowledge

E-mail address: christoph.wernhard@tu-dresden.de.

0747-7171/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2011.12.034

representation techniques in a unified way. This integration on the system level stems from the integration on the "theoretical" semantic framework level, which in turn could benefit from support by mechanized reasoning in the long run.

Work that has been done on the problem includes especially the development of circumscription (McCarthy, 1980), a semantically founded approach to non-monotonic reasoning that was devised as an add-on to first-order logic for knowledge representation. Properties and variants of circumscription have been investigated, e.g., in (Lifschitz, 1994, 1986). The expression of circumscription in terms of second-order quantification, and the corresponding processing by second-order quantifier elimination has been worked out (Doherty et al., 1997). Second-order quantifier elimination can also express further knowledge representation techniques such as abduction and modularization of knowledge bases (Gabbay et al., 2008). Variants of second-order quantifier elimination appear in the literature under different names such as *computation of uniform interpolants*, *forgetting*, and *projection*. Restricted to propositional formulas, it is called *elimination of quantified Boolean variables*. Our work is based on a particular variant of second-order quantification, *literal projection*, which permits, so to speak, to quantify upon an *arbitrary set of ground literals*, instead of just (all ground literals with) a given predicate. Literal projection allows, for example, to express predicate quantification upon a predicate just in positive or negative polarity. Eliminating such a quantifier from a formula in negation normal form results in a formula that might still contain the quantified predicate, but only in literals whose polarity is complementary to the quantified one. Literal projection has been specified originally for propositional logic (Lang et al., 2003) in its variant *literal forgetting*. This characterization has been reformulated in a more accessible way and generalized to first-order logic (Wernhard, 2008). Some relationships between literal projection and circumscription have been investigated for propositional logic (Lang et al., 2003).

The main contribution of the paper is a semantic framework that extends first-order logic by two semantically defined primitive operators, for literal projection, which we also call briefly *projection* from now on, and for *raising*. The latter is only slightly different from projection and can be applied to express the constraints imposed by circumscription to the models of the circumscribed formula. With these primitives, further operators are defined in compact ways, in particular for a generalization of parallel predicate circumscription with varied predicates, and for new variants of the strongest necessary and weakest sufficient condition (Lin, 2001; Doherty et al., 2001).

We call the variant of circumscription that is defined in terms of raising *scope-determined*, since like projection and raising its effects are controlled by a so-called *scope*, that is, a set of ground literals, as parameter. In the paper, a toolkit of propositions about projection, raising and circumscription and their interaction is presented. It reveals some refinements of and new views on previously known properties. In particular, a characterization of well-foundedness with respect to circumscription (Lifschitz, 1994) in terms of projection, and a characterization of consequences of circumscribed formulas (Lang et al., 2003) adapted to first-order logic and expressed compactly in terms of variants of the strongest necessary and weakest sufficient condition.

The toolkit of definitions and propositions provides a basis for formalizing applications that involve the interaction of projection and circumscription. It has been used in (Wernhard, 2010a) to characterize different established semantics for logic programming. An envisaged application area is the development of variants of such semantics to model human reasoning according to the approach of (Stenning and van Lambalgen, 2008; Hölldobler et al., 2011).

The originality of the contribution of the paper stems from two main novelties: first, the observation that circumscription can be characterized with raising whose semantic definition can be obtained from that of literal projection just by replacing a subset symbol with *strict* subset. Second, the idea to use scopes, sets of literals, as uniform parameters to control the effects of projection as well as circumscription.

The contribution is non-trivial in particular since it takes over the work of shifting the semantic operation of circumscription and projection, which is sometimes hard on the edge of intuition, to representations that are relatively easy to manipulate at the symbolic level, basically subset relationships between sets of literals. The contribution strives for simplicity and general applicability by consequently focusing on "semantic operators", operators that have equivalent values for equivalent argument formulas, as far as possible.

The paper is structured as follows: notation and preliminaries are given in Section 2. In Section 3 the concept of *scope* is introduced and background material on projection is provided. Section 4 is about scope-determined circumscription: the underlying raising operator is introduced (Section 4.1), the relationship of scope-determined circumscription to the traditional specification of predicate circumscription is shown (Section 4.2), and properties of scope-determined circumscription are developed, basic properties (Section 4.3) and properties that concern the interplay with projection (Sections 4.4 and 4.5). Variants of strongest necessary and weakest sufficient condition are introduced in Section 4.6 and applied to characterize consequences of circumscriptions in Section 4.7. Section 4.8 concludes Section 4 with a summary table of the propositions developed in its subsections. The conclusion (Section 5) is followed by an appendix with proofs of the propositions stated in Section 4.7.

Some of the material in this paper has been previously presented at the *7th International Workshop on First-Order Theorem Proving, FTP'09* (Wernhard, 2010b).

2. Notation and preliminaries

2.1. Symbolic notation

We use the following symbols, also with sub- and superscripts, to stand for items of types as indicated in the following table (precise definitions of the types are given later on), considered implicitly as universally quantified in definition and proposition statements:

F, G, H	–	Formula
A	–	Atom
L	–	Literal
S	–	Set of ground literals (also called *scope*)
I, J	–	Structure
β	–	Variable assignment

We write the positive (negative, resp.) *literal* with atom A as $+A$ ($-A$, resp.). We say that $+A$ ($-A$, resp.) has *positive* (*negative*, resp.) *polarity*. The *complement* of literal L is written $\overline{L}$. The *set of complements* of a set S of literals, that is, $\{\overline{L} | L \in S\}$, is written $\overline{S}$.

In writing sets of literals we use the following shorthands: a *predicate symbol* stands for the set of all ground literals with that symbol as predicate. A *set* or *tuple of predicate symbols* stands for the set of all ground literals whose predicate is a member of the set or tuple, respectively.

We assume a fixed first-order signature with at least one constant. The sets of all ground terms, all ground literals, all positive ground literals, and all negative ground literals – with respect to this signature – are denoted by TERMS, ALL, POS, NEG, respectively. *Variables* are x, y, z, also with subscripts.

2.2. Plain formulas

We assume that a *formula* is constructed from first-order literals and the logic operators shown in the left column of Table 1 (p. 1092). Later on we extend the notion of *formula* by two additional operators, for projection and raising, that go beyond first-order logic. To distinguish formulas without these operators, we call them *plain formulas*.

As meta-level notation with respect to this syntax, we use versions of the binary connectives with arbitrary integers ≥ 0 as arity, implication ($\rightarrow$), converse implication ($\leftarrow$), equivalence ($\leftrightarrow$), writing positive literals in formulas just as atoms, sequences of variables as quantifier arguments, and omitting of universal quantifiers. A *sentence* is a formula without free variables.

2.3. Semantic framework

We use a notational variant of the framework of Herbrand interpretations: an *interpretation* is a pair $\langle I, \beta \rangle$, where I is a *structure*, that is, a set of ground literals that contains for all ground atoms A

Table 1
The satisfaction relation for plain formulas.

$\langle I, \beta\rangle \models L$	iff_{def}	$L\beta \in I$.
$\langle I, \beta\rangle \models \top$.		
$\langle I, \beta\rangle \not\models \bot$.		
$\langle I, \beta\rangle \models \neg F$	iff_{def}	$\langle I, \beta\rangle \not\models F$.
$\langle I, \beta\rangle \models F_1 \wedge F_2$	iff_{def}	$\langle I, \beta\rangle \models F_1$ and $\langle I, \beta\rangle \models F_2$.
$\langle I, \beta\rangle \models F_1 \vee F_2$	iff_{def}	$\langle I, \beta\rangle \models F_1$ or $\langle I, \beta\rangle \models F_2$.
$\langle I, \beta\rangle \models \forall x\, F$	iff_{def}	for all $t \in \mathsf{TERMS}$ it holds that $\langle I, \beta\frac{t}{x}\rangle \models F$.
$\langle I, \beta\rangle \models \exists x\, F$	iff_{def}	there exists a $t \in \mathsf{TERMS}$ such that $\langle I, \beta\frac{t}{x}\rangle \models F$.

exactly one of $+A$ or $-A$, and β is a *variable assignment*, that is, a mapping of the set of variables into TERMS. Formula F with all free variables replaced by their image in β is denoted by $F\beta$; the variable assignment that maps x to ground term t and all other variables to the same values as β is denoted by $\beta\frac{t}{x}$.

The satisfaction relation between interpretations and formulas is defined by the clauses in Table 1, where L matches a literal, and F, F_1, F_2 a formula. A formula F is called *satisfiable* if and only if there exists an interpretation $\langle I, \beta\rangle$ such that $\langle I, \beta\rangle \models F$. Entailment and equivalence are straightforwardly defined in terms of the satisfaction relation. Entailment: $F_1 \models F_2$ holds if and only if for all $\langle I, \beta\rangle$ such that $\langle I, \beta\rangle \models F_1$ it holds that $\langle I, \beta\rangle \models F_2$. Equivalence: $F_1 \equiv F_2$ if and only if $F_1 \models F_2$ and $F_2 \models F_1$.

2.4. Relation to conventional model theory

The motivation for using sets of ground literals as the structure component of interpretations is that this facilitates the characterization of the extensions of first-order logic to be discussed in the subsequent sections. Relevant properties of structures can be expressed in a streamlined way as relationships of sets. However, interpretations according to our semantic framework can be just considered as a particular representation of interpretations as conventionally used in model theory: the set of literals I in an interpretation $\langle I, \beta\rangle$ is called *"structure"*, since it represents a Herbrand structure. The domain is the set of ground terms. Function symbols f with arity $n \geq 0$ are mapped to functions f' such that for all ground terms $t_1, \ldots, t_n$ it holds that $f'(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. Predicate symbols p with arity $n \geq 0$ are mapped to $\{\langle t_1, \ldots, t_n\rangle \mid +p(t_1, \ldots, t_n) \in I\}$. (We speak here explicitly of function *symbols* and predicate *symbols*, which we call in the other sections also briefly *functions* and *predicates*.) Moreover, an interpretation $\langle I, \beta\rangle$ represents a conventional second-order interpretation (Ebbinghaus et al., 1984) (if predicate variables are considered as distinguished predicate symbols): the structure in the conventional sense corresponds to I, as described above, except that mappings of predicate variables are omitted. The assignment is β, extended such that all predicate variables p are mapped to $\{\langle t_1, \ldots, t_n\rangle \mid +p(t_1, \ldots, t_n) \in I\}$.

3. Scopes and projection

3.1. Scopes

As already indicated, we will extend the notion of formula by two further primitive operators that go beyond first-order logic: $\mathsf{project}_S(F)$ for *projection* and $\mathsf{raise}_S(F)$ for *raising*. The arguments of both operators are a formula F and, written as subscript, a specifier of a set S of ground literals. We call a set of ground literals in the role as argument to these operators a *scope*. Projection and raising are then parametrized in the same way, with scopes providing a uniform interface for combining these operators and further operators defined in terms of them, like circumscription, as we will see later on. Scopes control the precise effects of operators with a granularity that goes down to the level of single ground literals: effects on a whole predicate can be expressed with scopes that contain all or none of the ground literals with the predicate; effects on a specific ground atom can be expressed with the

two-element set of its positive and negative literal; and effects that differ depending on the polarity in which atoms occur in the argument formula can be expressed by having just positive or negative literals in the scope.

We do not define here a concrete syntax for scope specifiers as formula constituents and just speak of a *scope*, referring to the actual scope in a semantic context as well as some expression that denotes it in a syntactic context. When writing scopes we make use of the shorthands specified in Section 2.1 for sets of literals.

3.2. Projection

The first of the two primitive operators by which we extend first-order logic is projection. Each of the standard operators for first-order logic has been semantically defined by a clause in Table 1. The semantic definition of projection provides such a clause for the projection operator:

Definition 1 (*Projection*)**.** The *projection* of formula F onto scope S, in symbols $\mathsf{project}_S(F)$, is a formula whose semantics is defined by

$$\langle I, \beta\rangle \models \mathsf{project}_S(F) \quad \text{iff}_{\text{def}} \quad \begin{array}{l}\text{there exists a } J \text{ such that}\\ \langle J, \beta\rangle \models F \text{ and } J \cap S \subseteq I.\end{array}$$

Forgetting is a notational variant of projection, where the scope is considered complementary. We define it here not as a primitive but in terms of projection:

Definition 2 (*Forgetting*)**.** The *forgetting* in formula F about scope S is defined as

$$\mathsf{forget}_S(F) \stackrel{\text{def}}{=} \mathsf{project}_{\mathsf{ALL}-S}(F).$$

Combined with propositional logic, projection generalizes Boolean quantification, combined with first-order logic second-order quantification: the second-order formula $\exists p\, F$, where p is a predicate symbol, can be expressed as projection of F onto the set of all ground literals with a predicate other than p, or equivalently, as the forgetting about the set of all ground literals with predicate p. Intuitively, the projection of a formula F onto scope S is a formula that expresses about literals in S the same as F, but expresses nothing about other literals. A projection of a plain propositional formula is equivalent to a plain propositional formula in negation normal form in which only literals in the projection scope do occur. Such a sentence is a *uniform interpolant* of the formula with respect to the scope. A naive way to construct such a sentence is indicated by the following equivalences, which hold for propositional formulas F and atoms A, where $F[A \mapsto \top]$ ($F[A \mapsto \bot]$, resp.) denotes F with all occurrences of atom A replaced by $\top$ ($\bot$, resp.):

$$\mathsf{forget}_{\{A\}}(F) \equiv F[A \mapsto \top] \vee F[A \mapsto \bot]. \tag{i}$$

$$\mathsf{forget}_{\{+A\}}(F) \equiv F[A \mapsto \top] \vee (\neg A \wedge F[A \mapsto \bot]). \tag{ii}$$

$$\mathsf{forget}_{\{-A\}}(F) \equiv (A \wedge F[A \mapsto \top]) \vee F[A \mapsto \bot]. \tag{iii}$$

The particular variants of projection and forgetting that we use are *literal projection* and *literal forgetting* (Wernhard, 2008; Lang et al., 2003), which allow, so-to-speak, to express quantification upon just the positive or negative occurrences of a predicate in a formula. They can be contrasted with *atom projection* and *atom forgetting*, respectively, where the polarity of the scope members is not taken into account.

Atom projection and atom forgetting can be defined as special cases of literal projection and literal forgetting, respectively, where the scope is constrained to be an *atom scope*, that is, containing the same atoms in positive as well as negative polarity:

Definition 3 (*Atom Scope*)**.** A scope S such that $S = \overline{S}$ is called an *atom scope*.

If S is an atom scope, the condition $J \cap S \subseteq I$ in the semantic definition of project just expresses that structures I and J are required to be equal as far as members of S are considered, but are unrelated otherwise:

Proposition 4 (*Structures Coinciding for Atom Projection*)**.** *If S is an atom scope, then*

$$J \cap S \subseteq I \quad \textit{if and only if} \quad J \cap S = I \cap S.$$

An example for atom projection is given with Example 5.i below. If S is not constrained to be an atom scope, the condition $J \cap S \subseteq I$ encodes a different effect on literals depending on whether they are positive or negative, as illustrated for q in Example 5.ii–5.v.

Example 5 (*Projection*)**.** Let $F \stackrel{\text{def}}{=} (-\mathsf{p} \vee +\mathsf{q}) \wedge (-\mathsf{q} \vee +\mathsf{r})$. Then

(i) $\mathsf{project}_{\{+\mathsf{p},-\mathsf{p},+\mathsf{r},-\mathsf{r}\}}(F) \equiv \mathsf{forget}_{\{+\mathsf{q},-\mathsf{q}\}}(F) \equiv -\mathsf{p} \vee +\mathsf{r}.$

(ii) $\mathsf{project}_{\{+\mathsf{p},-\mathsf{p},+q,+\mathsf{r},-\mathsf{r}\}}(F) \equiv \mathsf{forget}_{\{-\mathsf{q}\}}(F) \equiv (-\mathsf{p} \vee +\mathsf{q}) \wedge (-\mathsf{p} \vee +\mathsf{r}).$

(iii) $\mathsf{project}_{\{+\mathsf{p},-\mathsf{p},-q,+\mathsf{r},-\mathsf{r}\}}(F) \equiv \mathsf{forget}_{\{+\mathsf{q}\}}(F) \equiv (-\mathsf{q} \vee +\mathsf{r}) \wedge (-\mathsf{p} \vee +\mathsf{r}).$

(iv) $\mathsf{project}_{\{+\mathsf{p},-\mathsf{p},+q\}}(F) \equiv \mathsf{forget}_{\{-\mathsf{q},+\mathsf{r},-\mathsf{r}\}}(F) \equiv -\mathsf{p} \vee +\mathsf{q}.$

(v) $\mathsf{project}_{\{+\mathsf{p},-\mathsf{p},-q\}}(F) \equiv \mathsf{forget}_{\{+\mathsf{q},+\mathsf{r},-\mathsf{r}\}}(F) \equiv \top.$

Some of the properties of projection that we list in the proposition below involve the *literal base* of a plain formula, as we call the set of ground instances of the literals that "occur" in the formula. This is one of the few "syntactic" operators, that is, operators whose value might differ for equivalent formulas, which we will use. A related concept that is independent of syntactic properties, the *essential literal base*, is discussed in relation to projection in (Wernhard, 2008, 2009a). *Literal base* is formally defined as follows:

Definition 6 (*Literal Base*)**.** The *literal base* $\mathcal{L}(F)$ of a plain formula F is defined as follows: $\mathcal{L}(L)$ is the set of all ground instances of L; $\mathcal{L}(\top) \stackrel{\text{def}}{=} \mathcal{L}(\bot) \stackrel{\text{def}}{=} \{\}$; $\mathcal{L}(\neg F) \stackrel{\text{def}}{=} \overline{\mathcal{L}(F)}$; $\mathcal{L}(F_1 \wedge F_2) \stackrel{\text{def}}{=} \mathcal{L}(F_1 \vee F_2) \stackrel{\text{def}}{=} \mathcal{L}(F_1) \cup \mathcal{L}(F_2)$; $\mathcal{L}(\forall x F) \stackrel{\text{def}}{=} \mathcal{L}(\exists x F) \stackrel{\text{def}}{=} \mathcal{L}(F)$.

The following proposition gives an overview on properties of projection. Most of them follow straightforwardly from the semantic definition of project. Proofs, as well as more thorough material on projection can be found in (Wernhard, 2008, 2009a).

Proposition 7 (*Properties of Projection*)**.**

Basic Properties

(i) $F \models \mathsf{project}_S(F)$.

(ii) *If* $F_1 \models F_2$, *then* $\mathsf{project}_S(F_1) \models \mathsf{project}_S(F_2)$.

(iii) *If* $F_1 \equiv F_2$, *then* $\mathsf{project}_S(F_1) \equiv \mathsf{project}_S(F_2)$.

(iv) *If* $S_1 \supseteq S_2$, *then* $\mathsf{project}_{S_1}(F) \models \mathsf{project}_{S_2}(F)$.

(v) $\mathsf{project}_{S_2}(\mathsf{project}_{S_1}(F)) \equiv \mathsf{project}_{S_1 \cap S_2}(F)$.

(vi) $F_1 \models \mathsf{project}_S(F_2)$ *iff* $\mathsf{project}_S(F_1) \models \mathsf{project}_S(F_2)$.

(vii) $\mathsf{project}_{\mathsf{ALL}}(F) \equiv F$.

(viii) *F is satisfiable iff* $\mathsf{project}_S(F)$ *is satisfiable.*

(ix) *If no instance of L is in S, then* $\mathsf{project}_S(L) \equiv \top$.

(x) *If all instances of L are in S, then* $\mathsf{project}_S(L) \equiv L$.

Interplay with the Literal Base

(xi) *If F is plain, then* $\mathsf{project}_{\mathcal{L}(F)}(F) \equiv F$.

(xii) *If F is plain, then* $\mathsf{project}_S(F) \equiv \mathsf{project}_{\mathcal{L}(F)\cap S}(F)$.

(xiii) *If F_2 is plain, then $F_1 \models F_2$ iff* $\mathsf{project}_{\mathcal{L}(F_2)}(F_1) \models F_2$.

Interplay with Other Operators

(xiv) $\mathsf{project}_S(\top) \equiv \top$.

(xv) $\mathsf{project}_S(\bot) \equiv \bot$.

(xvi) $\mathsf{project}_S(F_1 \vee F_2) \equiv \mathsf{project}_S(F_1) \vee \mathsf{project}_S(F_2)$.

(xvii) $\mathsf{project}_S(F_1 \wedge F_2) \models \mathsf{project}_S(F_1) \wedge \mathsf{project}_S(F_2)$.

(xviii) *If F_1, F_2 are plain and $\mathcal{L}(F_1) \cap \overline{\mathcal{L}(F_2)} \subseteq S \cap \overline{S}$, then* $\mathsf{project}_S(F_1 \wedge F_2) \equiv \mathsf{project}_S(F_1) \wedge \mathsf{project}_S(F_2)$.

(xix) $\mathsf{project}_S(\exists x F) \equiv \exists x\, \mathsf{project}_S(F)$.

(xx) $\mathsf{project}_S(\forall x F) \models \forall x\, \mathsf{project}_S(F)$.

(xxi) $\mathsf{project}_{\overline{S}}(\neg \mathsf{project}_S(F)) \equiv \neg \mathsf{project}_S(F)$.

4. Scope-determined circumscription

4.1. The raising operator

The operator raise is, aside of project, the other "nonstandard" primitive by which we extend first-order logic. Analogously to projection it is semantically defined by a clause following the pattern in Table 1 for the first-order operators:

Definition 8 (*Raising*)**.** The *raising* of formula F onto scope S, in symbols $\mathsf{raise}_S(F)$, is a formula whose semantics is defined by

$$\langle I, \beta\rangle \models \mathsf{raise}_S(F) \quad \text{iff}_{\text{def}} \quad \begin{array}{l}\text{there exists a } J \text{ such that}\\ \langle J, \beta\rangle \models F \text{ and } J \cap S \subset I \cap S.\end{array}$$

When negated, the raise operator expresses the requirements that are additionally imposed by circumscription to the models of the circumscribed formula. Accordingly, we define a variant of predicate circumscription, *scope-determined circumscription*, in terms of raise:

Definition 9 (*Scope-determined circumscription*)**.** The *scope-determined circumscription* of formula F onto scope S, in symbols $\mathsf{circ}_S(F)$, is defined as

$$\mathsf{circ}_S(F) \stackrel{\text{def}}{=} F \wedge \neg \mathsf{raise}_S(F).$$

We will take a closer look on scope-determined circumscription in subsequent sections, and proceed for now with considering raise as an operator on its own. The semantic definitions of raise (Definition 8) and project (Definition 1) are very similar: the condition $J \cap S \subseteq I$ in the definition of project is equivalent to $J \cap S \subseteq I \cap S$. Just by replacing the subset relation ($\subseteq$) with *strict* subset ($\subset$), the definition of raise is obtained. The name *raising* refers to the requirement of the existence of a "lower" interpretation imposed by raise: an interpretation $\langle I, \beta\rangle$ is a model of $\mathsf{raise}_S(F)$ if and only if there exists another interpretation $\langle J, \beta\rangle$ which is a model of F and is strictly "lower" than $\langle I, \beta\rangle$ in the sense that $J \cap S \subset I \cap S$. In the definition of circumscription the raise operator occurs negated, thus ensuring that only "lowest" interpretations are models of a circumscription.

An alternate semantic characterization of raise provides further intuitive insight into its effect: a scope can be partitioned into two disjoint subsets which we call *biscope* and *uniscope*. The first contains those members of the scope whose complement is also a member of the scope (thus they are "*bi*-polar" members). The latter contains the remaining members of the scope, that is, those whose

complement is not also a member of the scope (thus they are "*uni*-polar" members). The following definition provides formal notation for partitioning a scope in this way:

Definition 10 (*Biscope and Uniscope Partitions of a Scope*)**.**

(i) $\mathsf{biscope}(S) \stackrel{\text{def}}{=} S \cap \overline{S}$.

(ii) $\mathsf{uniscope}(S) \stackrel{\text{def}}{=} S - \overline{S}$.

The semantic characterization of raise in the following proposition is like Definition 8, except that the condition $J \cap S \subset I \cap S$ is replaced by a condition that reveals the different effects of raise on members of the biscope and uniscope partitions of the raising scope: with respect to the biscope the structure J must be identical to I, and with respect to the uniscope it must be a strict subset of I.

Proposition 11 (*Raising in Terms of Biscopes and Uniscopes*)**.**

$$\langle I, \beta \rangle \models \mathsf{raise}_S(F)$$

if and only if there exists a J *such that*

(1) $\langle J, \beta \rangle \models F$,
(2) $J \cap \mathsf{biscope}(S) = I \cap \mathsf{biscope}(S)$, *and*
(3) $J \cap \mathsf{uniscope}(S) \subset I \cap \mathsf{uniscope}(S)$.

The following example shows the effect of raising for the same formula and scopes as Example 5 for projection:

Example 12 (*Raising*)**.** As in Example 5, let $F \stackrel{\text{def}}{=} (-\mathsf{p} \vee +\mathsf{q}) \wedge (-\mathsf{q} \vee +\mathsf{r})$. Then

(i) $\mathsf{raise}_{\{+\mathsf{p},-\mathsf{p},+\mathsf{r},-\mathsf{r}\}}(F) \equiv \bot$.

(ii) $\mathsf{raise}_{\{+\mathsf{p},-\mathsf{p},+\mathsf{q},+\mathsf{r},-\mathsf{r}\}}(F) \equiv -\mathsf{p} \wedge +\mathsf{q}$.

(iii) $\mathsf{raise}_{\{+\mathsf{p},-\mathsf{p},-\mathsf{q},+\mathsf{r},-\mathsf{r}\}}(F) \equiv -\mathsf{q} \wedge +\mathsf{r}$.

(iv) $\mathsf{raise}_{\{+\mathsf{p},-\mathsf{p},+\mathsf{q}\}}(F) \equiv -\mathsf{p} \wedge +\mathsf{q}$.

(v) $\mathsf{raise}_{\{+\mathsf{p},-\mathsf{p},-\mathsf{q}\}}(F) \equiv -\mathsf{q}$.

Properties of raising are compiled in the following proposition:

Proposition 13 (*Properties of Raising*)**.**

Basic Properties

(i) *If* $F_1 \models F_2$, *then* $\mathsf{raise}_S(F_1) \models \mathsf{raise}_S(F_2)$.

(ii) *If* $F_1 \equiv F_2$, *then* $\mathsf{raise}_S(F_1) \equiv \mathsf{raise}_S(F_2)$.

(iii) *If* $S_1 \supseteq S_2$ *and* $\mathsf{uniscope}(S_1) \subseteq \mathsf{uniscope}(S_2)$, *then* $\mathsf{raise}_{S_1}(F) \models \mathsf{raise}_{S_2}(F)$.

(iv) *If* $S = \overline{S}$, *then* $\mathsf{raise}_S(F) \equiv \bot$.

Interplay with Other Operators

(v) $\mathsf{raise}_S(F_1 \vee F_2) \equiv \mathsf{raise}_S(F_1) \vee \mathsf{raise}_S(F_2)$.

(vi) $\mathsf{raise}_S(F_1 \wedge F_2) \models \mathsf{raise}_S(F_1) \wedge \mathsf{raise}_S(F_2)$.

Interplay with Projection

(vii) $\mathsf{raise}_S(F) \models \mathsf{project}_S(F)$.

(viii) $\mathsf{project}_S(F) \equiv \mathsf{project}_{S \cup \overline{S}}(F) \vee \mathsf{raise}_S(F)$.

(ix) $\mathsf{project}_S(\mathsf{raise}_S(F)) \equiv \mathsf{raise}_S(F)$.

(x) *If* $S_c \subseteq S_p$, *then* $\mathsf{raise}_{S_c}(\mathsf{project}_{S_p}(F)) \equiv \mathsf{raise}_{S_c}(F)$.

(xi) *If* $S_p \subseteq S_c$ *and* $\mathsf{uniscope}(S_p) = \mathsf{uniscope}(S_c)$, *then* $\mathsf{raise}_{S_c}(\mathsf{project}_{S_p \cup \overline{S_p}}(F)) \equiv \mathsf{raise}_{S_p}(\mathsf{project}_{S_p \cup \overline{S_p}}(F))$.

Raising is monotonic (Proposition 13.i), like projection (Proposition 7.ii). From monotonicity follows that raise is a "semantic operator" in the sense that for equivalent argument formulas its values are also equivalent (Proposition 13.ii, analogous to Proposition 7.iii for project). A projection entails the projection onto a subset scope (Proposition 7.iv). The analog holds for raising with the additional precondition that the uniscope of the subset scope is a superset of the first scope (Proposition 13.iii). The raising onto an *atom* scope, or equivalently onto a scope with empty uniscope, is inconsistent (Proposition 13.iv).

Raising distributes over disjunction (Proposition 13.v), like projection (Proposition 7.xvi). The raising of a conjunction entails the raisings of its conjuncts (Proposition 13.vi), which follows from monotonicity and holds analogously also for projection (Proposition 7.xvii).

That raising entails projection (Proposition 13.vii) follows immediately from their semantic definitions. Proposition 13.viii is a stronger statement which shows that raising can be applied together with *atom* projection to characterize *literal* projection. Nestings of alternating applications of projection and raising onto the same scope collapse into just raising, since a raising does only express knowledge about the raising scope (Proposition 13.ix) and only the knowledge that a formula expresses about the raising scope is relevant for raising (Proposition 13.x). Proposition 13.xi underlies properties of circumscription discussed later on, specifically Proposition 18.iii and 18.iv.

4.2. Generalizing predicate circumscription

The scope-determined circumscription $\mathsf{circ}_S(F)$ of a formula F onto scope S has been defined in terms of the raise operator in Definition 9. It expresses a generalization of predicate circumscription (McCarthy, 1980). The attribute *scope-determined* indicates that a scope, that is, a set of ground literals, is used to determine what is circumscribed. If F is a plain sentence over disjoint sets of predicates P, Q and Z, then the *parallel predicate circumscription of P in F with fixed Q and varied Z* (Lifschitz, 1994), traditionally written $\mathsf{CIRC}[F; P; Z]$, can be expressed as $\mathsf{circ}_{(P\cap\mathsf{POS})\cup Q}(F)$. Recall that in specifications of scopes we let a set of predicates stand for the set of all ground instances of literals whose predicate is in the set (see Section 2.1). The scope $(P \cap \mathsf{POS}) \cup Q$ thus is the set of

(1) all *positive* ground literals with a circumscribed predicate, and
(2) *all* ground literals with a fixed predicate.

Scope-determined circumscription is more general than parallel predicate circumscription: Since a scope is an arbitrary set of ground literals, with scope-determined circumscription it is possible to express minimization, maximization and variation conditions that apply only to a subset of the instances of a predicate. The set of those ground instances of a predicate that are just positively in the circumscription scope forms the set of minimized instances of the predicate. Symmetrically, the instances that are just negatively in the scope form the set of maximized instances. The instances that are in both polarities in the scope form the set of fixed instances. The remaining instances, that is, those ground instances of the predicate that are neither positively nor negatively contained in the scope, form the set of varied instances.

We now make precise how scope-determined circumscription relates to the established definition of predicate circumscription by means of second-order quantification (Lifschitz, 1994; Doherty et al., 1997; Gabbay et al., 2008). We express the involved second-order quantification by projection, justified as follows: $\exists p\, F$ can be expressed as $\mathsf{project}_S(F)$, where S is the set of all ground literals with a predicate other than p. From Proposition 7.xii it can be derived that also a smaller projection scope is sufficient: $\mathsf{project}_S(F)$ is equivalent to $\mathsf{project}_{S'}(F)$ for all subsets S' of S that contain those literals of S whose predicate symbols actually occur in F. The above-mentioned parallel circumscription traditionally written as $\mathsf{CIRC}[F; P; Z]$, which is equivalent to the sentence called *second-order circumscription of P in F with variable Z* in (Doherty et al., 1997; Gabbay et al., 2008), can then be notated as the sentence $\mathsf{circ\text{-}so}(F, P, Z)$ specified as follows:

Definition 14 (*Second-Order Circumscription in Terms of Projection*)**.** Let F be a plain first-order sentence and let $P = \langle p_1, \ldots, p_n\rangle$ and $Z = \langle z_1, \ldots, z_m\rangle$ with $n, m \geq 0$ be disjoint tuples of distinct

predicate symbols that occur in F. Then $\mathsf{circ\text{-}so}(F, P, Z)$ is a sentence with the projection operator, defined as:

$$\mathsf{circ\text{-}so}(F, P, Z) \stackrel{\text{def}}{=} F \wedge \neg \mathsf{project}_{P \cup Q}(F' \wedge P' < P),$$

where Q is the set of ground literals whose predicate symbol occurs in F but is neither in P nor in Z, and F' and $P' < P$ are defined as follows: Let $P' = \langle p'_1, \ldots, p'_n \rangle$ and $Z' = \langle z'_1, \ldots, z'_m \rangle$ be disjoint tuples of distinct predicate symbols such that members of P' and P with the same index, as well as members of Z' and Z with the same index, are of the same arity, and P' and Z' do not contain predicate symbols in F. Let F' be the formula that is obtained from F by replacing each predicate symbol that is in P or Z by the predicate symbol with the same index in P' or Z', respectively. For $i \in \{1, \ldots, n\}$ let $\boldsymbol{x}_i$ stand for $x_1, \ldots, x_k$, where k is the arity of predicate symbol p_i. Finally, let $P' < P$ stand for

$$\bigwedge_{i=1}^{n} \forall \boldsymbol{x}_i (p'_i(\boldsymbol{x}_i) \rightarrow p_i(\boldsymbol{x}_i)) \wedge \neg \bigwedge_{i=1}^{n} \forall \boldsymbol{x}_i (p'_i(\boldsymbol{x}_i) \leftrightarrow p_i(\boldsymbol{x}_i)).$$

The Q parameter on the right side of Definition 14 represents the "fixed" predicate symbols. The set of literals $P \cup Q$ suffices as projection scope to "forget about" P' and Z', since the projected formula $F' \wedge P' < P$, contains, aside of predicates symbols in P', Z', just predicates symbols that are in P or in Q. The following proposition states formally that second-order circumscription can be expressed with scope-determined circumscription.

Proposition 15 (*Second-Order and Scope-Determined Circumscription*)**.** *Let F be a plain first-order sentence and let P, Z be tuples of predicate symbols as specified in the definition of* circ-so*. Let Q be the set of predicate symbols in F that are neither in P nor in Z. Then*

$$\mathsf{circ\text{-}so}(F, P, Z) \equiv \mathsf{circ}_{(P \cap \mathsf{POS}) \cup Q}(F).$$

Pointwise circumscription (Lifschitz, 1986) can be expressed as conjunction over scope-determined circumscriptions onto scopes of a specific form: If S is an atom scope and L is a positive literal in S, then the circumscription onto the scope $S - \{\overline{L}\}$ can be understood as circumscribing the "single point" L. If F is a sentence over disjoint sets of predicates P, Q and Z, then the pointwise counterpart to $\mathsf{circ}_{(P \cap \mathsf{POS}) \cup Q}(F)$ is $\bigwedge_{L \in P \cap \mathsf{POS}} \mathsf{circ}_{(P - \overline{L}) \cup Q}(F)$, where, to let the conjunction be finite, we assume that P has a finite set of ground instances.

4.3. Basic properties of scope-determined circumscription

The following proposition lists some properties of scope-determined circumscription that do not involve other logic operators and follow from its definition and properties of raising:

Proposition 16 (*Basic Properties of Scope-Determined Circumscription*)**.**

(i) $\mathsf{circ}_S(F) \models F$.

(ii) *If* $F_1 \equiv F_2$, *then* $\mathsf{circ}_S(F_1) \equiv \mathsf{circ}_S(F_2)$.

(iii) *If* $S_1 \subseteq S_2$ *and* $\mathsf{uniscope}(S_1) \supseteq \mathsf{uniscope}(S_2)$, *then* $\mathsf{circ}_{S_1}(F) \models \mathsf{circ}_{S_2}(F)$.

(iv) *If* $S_1 \subseteq S_2$ *and* $\mathsf{uniscope}(S_1) \supseteq \mathsf{uniscope}(S_2)$, *then* $\mathsf{circ}_{S_2}(\mathsf{circ}_{S_1}(F)) \equiv \mathsf{circ}_{S_1}(F)$.

(v) *If* $S = \overline{S}$, *then* $\mathsf{circ}_S(F) \equiv F$.

A circumscription entails its argument formula (Proposition 16.i). Although not monotonic, circumscription is, like raising and projection, a "semantic operator" in the sense that for equivalent argument formulas its values are also equivalent (Proposition 16.ii). A circumscription entails the circumscription onto a superset scope, provided that its uniscope is a subset of the first

scope (Proposition 16.iii). If this relationship holds between inner and outer scopes of nested circumscriptions, they collapse into a circumscription onto the inner scope (Proposition 16.iv). The circumscription onto an *atom* scope, or equivalently onto a scope with empty uniscope, is equivalent to the circumscribed formula (Proposition 16.v). The following facts about the interplay of scope-determined circumscription with disjunction and conjunction also follow from properties of raising:

Proposition 17 (*Interplay of Circumscription with Disjunction and Conjunction*)**.**

(i) $\mathsf{circ}_S(F_1 \vee F_2) \models \mathsf{circ}_S(F_1) \vee \mathsf{circ}_S(F_2)$.

(ii) $\mathsf{circ}_S(F_1) \wedge \mathsf{circ}_S(F_2) \models \mathsf{circ}_S(F_1 \wedge F_2)$.

The circumscription of a disjunction entails the disjunction of the circumscriptions of its disjuncts (Proposition 17.i). A conjunction of circumscriptions entails the circumscription of the conjunction of their circumscribed formulas (Proposition 17.ii).

4.4. Circumscriptions of projections

The following proposition states properties of the circumscription operator wrapped around projection:

Proposition 18 (*Circumscriptions of Projections*)**.**

(i) *If* $S_c \subseteq S_p$, *then*
$\mathsf{circ}_{S_c}(F) \models \mathsf{circ}_{S_c}(\mathsf{project}_{S_p}(F))$.

(ii) $\mathsf{circ}_S(\mathsf{project}_S(F)) \equiv \mathsf{circ}_S(\mathsf{project}_{S \cup \overline{S}}(F))$.

(iii) *If* $S' \cap \overline{S} = \emptyset$ *and* $S' = \overline{S'}$, *then*
$\mathsf{circ}_S(F) \equiv F \wedge \mathsf{circ}_{S \cup S'}(\mathsf{project}_{S \cup \overline{S}}(F))$.

(iv) *If* $S_p \subseteq S_c$ *and* $\mathsf{uniscope}(S_p) = \mathsf{uniscope}(S_c)$, *then*
$\mathsf{circ}_{S_p}(\mathsf{project}_{S_p \cup \overline{S_p}}(F)) \equiv \mathsf{circ}_{S_c}(\mathsf{project}_{S_p \cup \overline{S_p}}(F))$.

(v) *Let* $S \stackrel{\text{def}}{=} S_p \cup \overline{\mathsf{uniscope}(S_c)} \cup (\mathsf{biscope}(S_c) \cap \overline{\mathsf{uniscope}(S_p)})$. *Then*
$\mathsf{circ}_{S_c}(\mathsf{project}_{S_p}(F)) \equiv \mathsf{project}_S(\mathsf{circ}_{S_c}(\mathsf{project}_{S_p}(F)))$.

A circumscription entails the circumscription of a projection onto a superset of the circumscription scope (Proposition 18.i). The circumscription of a projection, both onto the same scope, is equivalent to the circumscription onto this scope of the projection onto a possibly larger scope: the atom scope obtained as union of the scope and the set of complements of its members (Proposition 18.ii). In the special case where $F \equiv \mathsf{project}_{S \cup \overline{S}}(F)$, which holds if the circumscription scope S corresponds to predicate circumscription without varied predicates, the statement Proposition 18.ii is equivalent to

$$\mathsf{circ}_S(\mathsf{project}_S(F)) \equiv \mathsf{circ}_S(F). \tag{iv}$$

Predicate circumscription with fixed and varied predicates can be expressed in terms of predicate circumscription with just fixed predicates. This is rendered by equivalence (v) below in terms of scope-determined circumscription and atom projection. Let F be a sentence over disjoint sets of predicates P, Q and Z. As explained in Section 4.2, the parallel predicate circumscription of P in F with fixed Q and varied Z can then be expressed as the scope-determined circumscription $\mathsf{circ}_{(P \cap \mathsf{POS}) \cup Q}(F)$. By instantiating Proposition 18.iii with $S = (P \cap \mathsf{POS}) \cup Q$ and $S' = Z$ we obtain:

$$\mathsf{circ}_{(P \cap \mathsf{POS}) \cup Q}(F) \equiv \mathsf{circ}_{(P \cap \mathsf{POS}) \cup Q \cup Z}(\mathsf{project}_{P \cup Q}(F)). \tag{v}$$

Proposition 18.iv shows a condition for equivalence preserving adjustments of the scopes of circumscriptions by adding or removing "irrelevant" literals. For example, let P, Q be disjoint sets

of predicates and let F be a formula that is just over P. Then $F \equiv \mathsf{project}_P(F)$, and Proposition 18.iv can be instantiated with $S_p = P \cap \mathsf{POS}$ and $S_c = (P \cap \mathsf{POS}) \cup Q$ to derive

$$\mathsf{circ}_{P\cap\mathsf{POS}}(F) \equiv \mathsf{circ}_{(P\cap\mathsf{POS})\cup Q}(F). \tag{vi}$$

An equivalence of the form

$$F \equiv \mathsf{project}_S(F) \tag{vii}$$

can be understood as a statement that F only expresses something about literals in S. The equivalence of Proposition 18.v is of this form, specifying in this way a scope that includes all literals about which a circumscription does express something. The proposition can be read as stating that if the formula F which is going to be circumscribed is only about literals in S_p, that is, if it holds that $F \equiv \mathsf{project}_{S_p}(F)$, then the scope S includes all literals about which the circumscription onto scope S_c does express something. That is, the circumscription is equivalent to its projection onto S. Scope S is the union of three constituents:

(1) S_p: Literals about which the formula to be circumscribed might express something.
(2) $\overline{\mathsf{uniscope}(S_c)}$: Complements of literals in the uniscope of the circumscription scope. That is, complements of minimized or maximized literals.
(3) $\mathsf{biscope}(S_c) \cap \overline{\mathsf{uniscope}(S_p)}$: Literals in the biscope of the circumscription scope whose complement is contained in the uniscope of S_p. That is, literals which are fixed with respect to the circumscription, and about whose complement the circumscribed formula might express something (but does not express something about the literals themselves — such literals would already be included by item (1) in S).

These three constituents are illustrated by the following example:

Example 19 (*Aboutness of Circumscription*)**.** Let $F \stackrel{\text{def}}{=} \mathsf{p} \wedge (\mathsf{q} \rightarrow \mathsf{r})$, $S_p \stackrel{\text{def}}{=} \{+\mathsf{p}, -\mathsf{q}, +\mathsf{r}\}$, and $S_c \stackrel{\text{def}}{=} \{+\mathsf{q}, -\mathsf{q}, +\mathsf{r}, +\mathsf{s}\}$. Then $F \equiv \mathsf{project}_{S_p}(F)$ and

$$\mathsf{circ}_{S_c}(F) \equiv \mathsf{p} \wedge (\mathsf{r} \leftrightarrow \mathsf{q}) \wedge \neg\mathsf{s}. \tag{viii}$$

By Proposition 18.v it holds that $\mathsf{circ}_{S_c}(F) \equiv \mathsf{project}_S(\mathsf{circ}_{S_c}(F))$, where $S = S_p \cup \overline{\mathsf{uniscope}(S_c)} \cup (\mathsf{biscope}(S_c) \cap \overline{\mathsf{uniscope}(S_p)})$. The three constituents of S then are

(1) $S_p = \{+\mathsf{p}, -\mathsf{q}, +\mathsf{r}\}$,
(2) $\overline{\mathsf{uniscope}(S_c)} = \{-\mathsf{r}, -\mathsf{s}\}$, and
(3) $\mathsf{biscope}(S_c) \cap \overline{\mathsf{uniscope}(S_p)} = \{+\mathsf{q}\}$.

Their union is $S = \{+\mathsf{p}, +\mathsf{q}, -\mathsf{q}, +\mathsf{r}, -\mathsf{r}, -\mathsf{s}\}$. Since S is also the literal base of the right side of equivalence (viii), it follows from Proposition 7.xi that $\mathsf{circ}_{S_c}(F) \equiv \mathsf{project}_S(\mathsf{circ}_{S_c}(F))$, as claimed by Proposition 18.v.

4.5. Well-foundedness and projections of circumscriptions

As discussed in (Lifschitz, 1994), circumscription can in general only be applied usefully to a formula F if all models of F extend some model of F that is minimal with respect to the circumscribed predicates. In the extreme case where no model of F extends a minimal one, that is, if for all models of F there exists another one that is strictly smaller with respect to the circumscribed predicates, the circumscription is unsatisfiable. The property *well-founded*, which holds for universal formulas, makes the required condition precise. As presented in (Lifschitz, 1994), it is defined for circumscription of a single predicate p with varied predicates Z as follows (adapted to our notation): Let F be a plain first-order sentence, p be predicate symbol and Z be a tuple of predicate symbols. The sentence F is

called *well-founded* with respect to (p, Z) if for every model $\langle I, \beta\rangle$ of F there exists a model $\langle J, \beta\rangle$ of $\mathsf{circ\text{-}so}(F, \langle p\rangle, Z)$ such that $\langle I, \beta\rangle$ and $\langle J, \beta\rangle$ differ only in how they interpret p and Z and the extent of p in $\langle J, \beta\rangle$ is a (not necessarily strict) subset of its extent in $\langle I, \beta\rangle$. We can convert this definition straightforwardly into our semantic framework: Let Q be the set of predicate symbols in F that are different from p and not in Z. The sentence F is then well-founded with respect to (p, Z) if for all interpretations $\langle I, \beta\rangle$ such that $\langle I, \beta\rangle \models F$ there exists an interpretation $\langle J, \beta\rangle$ such that

(1) $\langle J, \beta\rangle \models \mathsf{circ\text{-}so}(F, \langle p\rangle, Z)$,
(2) $J \cap p \cap \mathsf{POS} \subseteq I$, and
(3) $J \cap Q = I \cap Q$.

The project operator allows to express this definition compactly for scope-determined circumscription: Let S be the scope $(p \cap \mathsf{POS}) \cup Q$. By Proposition 15, $\mathsf{circ\text{-}so}(F, \langle p\rangle, Z)$ is equivalent to $\mathsf{circ}_S(F)$. Furthermore, given that I and J are structures and $Q = \overline{Q}$, the conjunction of items (2) and (3) above is equivalent to $J \cap S \subseteq I$. By contracting the definition of project (Definition 1), the statement that there exists an interpretation $\langle J, \beta\rangle$ satisfying items (1)–(3) can be expressed as

$$\langle I, \beta\rangle \models \mathsf{project}_S(\mathsf{circ}_S(F)). \tag{ix}$$

Accordingly, the following definition provides a compact characterization of well-foundedness in terms of projection and scope-determined circumscription. It applies with respect to arbitrary scopes S, corresponding to variants of circumscription as indicated in Section 4.2:

Definition 20 (*Well-Founded Formula*)**.** A formula F is called *well-founded* with respect to scope S if and only if

$$F \models \mathsf{project}_S(\mathsf{circ}_S(F)).$$

The following proposition shows that for well-founded formulas the projection of a circumscription to a superset of the projection scope collapses to just the projection:

Proposition 21 (*Projections of Circumscriptions*)**.** *If F is well-founded with respect to S_c and $S_p \subseteq S_c$, then*

$$\mathsf{project}_{S_p}(\mathsf{circ}_{S_c}(F)) \equiv \mathsf{project}_{S_p}(F).$$

4.6. Globally strongest necessary and weakest sufficient condition

The properties of consequences of circumscription that will be discussed in the subsequent section involve the application of projection according to two specific related patterns, defined as follows:

Definition 22 (*Globally Strongest Necessary Condition*)**.** The *globally strongest necessary condition* of formula G on scope S within formula F, in symbols $\mathsf{gwsc}_S(F, G)$, is defined as

$$\mathsf{gsnc}_S(F, G) \stackrel{\text{def}}{=} \mathsf{project}_S(F \wedge G).$$

Definition 23 (*Globally Weakest Sufficient Condition*)**.** The *globally weakest sufficient condition* of formula G on scope S within formula F, in symbols $\mathsf{gwsc}_S(F, G)$, is defined as

$$\mathsf{gwsc}_S(F, G) \stackrel{\text{def}}{=} \neg\mathsf{project}_{\overline{S}}(F \wedge \neg G).$$

The following two propositions show alternate characterizations of these operators:

Proposition 24 (*Alternate Characterization of gsnc*)**.** *A formula H is equivalent to* $\mathsf{gsnc}_S(F, G)$ *if and only if the following holds:*

(1) $H \equiv \mathsf{project}_S(H)$.
(2) $F \models G \rightarrow H$.
(3) *For all formulas H' such that* $H' \equiv \mathsf{project}_S(H')$ *and* $F \models G \rightarrow H'$ *it holds that* $H \models H'$.

Proposition 25 (*Alternate Characterization of gwsc*)**.** *A formula H is equivalent to* $\mathsf{gwsc}_S(F, G)$ *if and only if the following holds:*

(1) $H \equiv \mathsf{project}_S(H)$.
(2) $F \models H \rightarrow G$.
(3) *For all formulas H' such that* $H' \equiv \mathsf{project}_S(H')$ *and* $F \models H' \rightarrow G$ *it holds that* $H' \models H$.

Globally strongest necessary condition and *globally weakest sufficient condition* are quite common patterns of the application of projection. For example, forms of abduction and notions of conservative theory extension can be expressed with instances of $\mathsf{gwsc}_S(F)$. These patterns are variants of *strongest necessary condition* and *weakest sufficient condition*, which have been devised in (Lin, 2001) for propositional logic and adapted to first-order logic in (Doherty et al., 2001). The variants introduced here differ in several details, which we point out now for the globally strongest necessary condition. This applies analogously for the globally weakest sufficient condition. The following definition renders *strongest necessary condition* as defined in (Doherty et al., 2001):

Definition 26 (*Strongest Necessary Condition*)**.** A strongest necessary condition of a formula G on a set of predicate symbols P under formula F is a formula H such that:

(1) H contains only predicate symbols from P.
(2) $F \models G \rightarrow H$.
(3) For all formulas H' such that H' contains only predicate symbols from P and $F \models G \rightarrow H'$ it holds that $F \models H \rightarrow H'$.

There are minor differences to the original definition which we neglect here: the F parameter is originally called a *theory* (seemingly referring to a set of formulas) instead of a formula. Also, the original definition introduces the additional intermediate concept *sufficient condition* for H satisfying conditions (1) and (2). The essential differences are the following:

(1) In the global variant a *scope* argument is used instead of a set of predicates to specify the symbols permitted in H. The involvement of scopes allows to constrain also the polarity in which predicates are allowed to occur in H. This feature is essential for Propositions 27.ii and 27.iii below. In addition, with scopes it can be specified that only particular instances of a predicate are allowed to occur in H.
(2) The restriction on the vocabulary of H is expressed in the global variant by a semantic condition ($H \equiv \mathsf{project}_S(H)$), that is, a condition which is independent of syntactic properties of H. In contrast, the strongest necessary condition refers to the predicate symbols contained in H, a syntactic property. For propositional logic, this difference is neglectable, since if a formula H is known to satisfy $H \equiv \mathsf{project}_S(H)$, a formula that is equivalent to H and only contains literals from S can always be constructed from H.
(3) The consequents in the respective conditions (3) of Proposition 24 and Definition 26 are different. For the global variant it is $H \models H'$, while, for the strongest necessary condition it is $F \models H \rightarrow H'$. This implies that for the same parameters there may exist strongest necessary conditions H_1, H_2 that are not equivalent, but equivalent under the precondition F, that is, $F \models H_1 \leftrightarrow H_2$. The following example from (Lin, 2001, Example 1.3) shows this: Let $F = \{(\mathsf{q} \rightarrow \mathsf{p}_1) \wedge \mathsf{q}\}$, $G = \mathsf{q}$ and $S = \{\mathsf{p}_1\}$. Then formulas p_1 and $\top$ are both strongest necessary conditions of G on S under F but clearly not equivalent. In contrast, the globally strongest necessary condition is unique up

to equivalence, and strongest compared to any H' satisfying the antecedent of condition (3) in Proposition 24, independently of F. Hence the prefix "*globally*" for the variants introduced here. Apparently, the only reason for basing the definition of *strongest necessary condition* on the relativized consequent $F \models H \rightarrow H'$ is a property that facilitates a certain computation technique (Lin, 2001). In (Doherty et al., 2001) both variants are not clearly distinguished: *Strongest necessary condition* with the relativized consequent is associated with a characterization in terms of second-order quantification that can be considered as instance of the *globally* strongest necessary condition.

4.7. Characterizing consequences of circumscription

Proposition 27 shows that a formula is a consequence of a circumscription if and only if – depending on the vocabulary of the formula – the formula itself, or a formula obtained by applying the globally strongest necessary and globally weakest sufficient condition is a consequence of the circumscribed formula:

Proposition 27 (*Consequences of Circumscriptions*)**.**

(i) *If F is well-founded with respect to S and $G \equiv \mathsf{project}_S(G)$, then* $\mathsf{circ}_S(F) \models G$ *iff* $F \models G$.

(ii) *If F is well-founded with respect to S and $G \equiv \mathsf{project}_{S\cup\overline{S}}(G)$, then* $\mathsf{circ}_S(F) \models G$ *iff* $F \models \mathsf{gsnc}_S(F, G)$.

(iii) *If F is well-founded with respect to S, then* $\mathsf{circ}_S(F) \models G$ *iff* $F \models \mathsf{gsnc}_S(F, \mathsf{gwsc}_{\overline{S}}(F, G))$.

Propositions 27.i–27.iii differ by trading-off simplicity of the characterization against strength of the precondition that restricts the consequence formula. These propositions generalize and combine adaptions of propositions in (Lifschitz, 1994; Lang et al., 2003): Proposition 27.ii and 27.iii are based on characterizations of the consequences of propositional circumscription in terms of literal forgetting given as Proposition 22 in (Lang et al., 2003), an investigation of propositional literal forgetting. The statements given in (Lang et al., 2003) are generalized and made more precise here in the following respects:

(1) Proposition 27.ii and 27.iii apply to first-order logic. The precondition that F is well-founded has been added since it is required to show these propositions for first-order logic in general. Propositional formulas are always well-founded.

(2) Preconditions that constrain the vocabulary of the consequences are expressed by statements $G \equiv \mathsf{project}_S(G)$ and $G \equiv \mathsf{project}_{S\cup\overline{S}}(G)$, in contrast to $\mathcal{L}(G) \subseteq S$ and $\mathcal{L}(G) \subseteq S \cup \overline{S}$, respectively. The statements with the projection operator are independent of syntactic properties of G and more general than those which refer to the syntactic literal base.

(3) Observing that projection is applied in patterns matching the globally strongest necessary and globally weakest sufficient condition, we apply these operators to express the properties more compactly.

(4) A thorough proof is provided in the appendix. The proof given in (Lang et al., 2003) just shows the characterizations as straightforward consequence of (Przymusinski, 1989, Theorems 2.5 and 2.6), for which in turn no proof is given, neither in (Przymusinski, 1989), nor in (Gelfond et al., 1986) which is referenced by (Przymusinski, 1989).

(5) With Proposition 27.i a third basic variant for consequents that are stronger restricted than in Proposition 27.ii is fitted in. This basic variant is actually a straightforward generalization of Proposition 12 in (Lifschitz, 1994), which is introduced as capturing the intuition that, under the assumption of well-foundedness, a circumscription provides no new information about the fixed predicates, and only "negative" additional information about the circumscribed predicates.

4.8. Summary of properties of scope-determined circumscription

We conclude the presentation of scope-determined circumscription with Table 2 (p. 1104) that displays the properties stated as propositions in the preceding sections all together at a single place.

Table 2
Summary of properties of scope-determined circumscription. Numbers indicate the respective propositions.

Basic properties

(16.i) $\mathsf{circ}_S(F) \models F$.

(16.ii) *If* $F_1 \equiv F_2$, *then* $\mathsf{circ}_S(F_1) \equiv \mathsf{circ}_S(F_2)$.

(16.iii) *If* $S_1 \subseteq S_2$ *and* $\mathsf{uniscope}(S_1) \supseteq \mathsf{uniscope}(S_2)$, *then* $\mathsf{circ}_{S_1}(F) \models \mathsf{circ}_{S_2}(F)$.

(16.iv) *If* $S_1 \subseteq S_2$ *and* $\mathsf{uniscope}(S_1) \supseteq \mathsf{uniscope}(S_2)$, *then* $\mathsf{circ}_{S_2}(\mathsf{circ}_{S_1}(F)) \equiv \mathsf{circ}_{S_1}(F)$.

(16.v) *If* $S = \overline{S}$, *then* $\mathsf{circ}_S(F) \equiv F$.

Interplay with other operators

(17.i) $\mathsf{circ}_S(F_1 \vee F_2) \models \mathsf{circ}_S(F_1) \vee \mathsf{circ}_S(F_2)$.

(17.ii) $\mathsf{circ}_S(F_1) \wedge \mathsf{circ}_S(F_2) \models \mathsf{circ}_S(F_1 \wedge F_2)$.

Circumscriptions of projections

(18.i) *If* $S_c \subseteq S_p$, *then* $\mathsf{circ}_{S_c}(F) \models \mathsf{circ}_{S_c}(\mathsf{project}_{S_p}(F))$.

(18.ii) $\mathsf{circ}_S(\mathsf{project}_S(F)) \equiv \mathsf{circ}_S(\mathsf{project}_{S\cup\overline{S}}(F))$.

Varied predicates via projection

(18.iii) *If* $S' \cap \overline{S} = \emptyset$ *and* $S' = \overline{S'}$, *then* $\mathsf{circ}_S(F) \equiv F \wedge \mathsf{circ}_{S\cup S'}(\mathsf{project}_{S\cup\overline{S}}(F))$.

Equivalent circumscription scopes

(18.iv) *If* $S_p \subseteq S_c$ *and* $\mathsf{uniscope}(S_p) = \mathsf{uniscope}(S_c)$, *then* $\mathsf{circ}_{S_p}(\mathsf{project}_{S_p\cup\overline{S_p}}(F)) \equiv \mathsf{circ}_{S_c}(\mathsf{project}_{S_p\cup\overline{S_p}}(F))$.

Aboutness of circumscription

(18.v) *Let* $S \stackrel{\text{def}}{=} S_p \cup \overline{\mathsf{uniscope}(S_c)} \cup (\mathsf{biscope}(S_c) \cap \overline{\mathsf{uniscope}(S_p)})$. *Then* $\mathsf{circ}_{S_c}(\mathsf{project}_{S_p}(F)) \equiv \mathsf{project}_S(\mathsf{circ}_{S_c}(\mathsf{project}_{S_p}(F)))$.

Projections of circumscriptions

(21) *If* F *is well-founded with respect to* S_c *and* $S_p \subseteq S_c$, *then* $\mathsf{project}_{S_p}(\mathsf{circ}_{S_c}(F)) \equiv \mathsf{project}_{S_p}(F)$.

Consequences of circumscriptions

(27.i) *If* F *is well-founded with respect to* S *and* $G \equiv \mathsf{project}_S(G)$, *then* $\mathsf{circ}_S(F) \models G$ *iff* $F \models G$.

(27.ii) *If* F *is well-founded with respect to* S *and* $G \equiv \mathsf{project}_{S\cup\overline{S}}(G)$, *then* $\mathsf{circ}_S(F) \models G$ *iff* $F \models \mathsf{gsnc}_S(F, G)$.

(27.iii) *If* F *is well-founded with respect to* S, *then* $\mathsf{circ}_S(F) \models G$ *iff* $F \models \mathsf{gsnc}_S(F, \mathsf{gwsc}_{\overline{S}}(F, G))$.

5. Conclusion

We have introduced the raising operator which can be used to define circumscription in a compact way. The semantic definitions of literal projection and raising can be written such that they differ only in that a set inclusion symbol in the definition of literal projection is in place of a strict set inclusion symbol in the definition of raising. The raising operator has – aside of a formula – just a so-called scope, that is, a set of literals, as argument, such that, depending on the composition of this

set, not only parallel circumscription with varied predicates can be expressed, but also minimization, maximization and variation conditions that apply only to a subset of the instances of a predicate.

The characterization of circumscription in terms of the raising operator is immediately useful to prove properties of circumscription in a streamlined way. Properties that involve circumscription together with projection can be straightforwardly expressed since operators for both are parametrized uniformly with scopes that can then be shared or related between operator occurrences. The introduced semantic framework, an extension of first-order logic by projection and raising, provides a basis for future research, including the further elaboration of common and differing properties of both operators, and the first-order based reconstruction of various knowledge representation techniques.

The detailed formalization with its relatively simple manageability at the symbolic level should facilitate mechanization, which can then be used to reason with machine support about combinations of projection and circumscription. For this, there are different approaches conceivable: first, on the "meta-level", by submitting the semantic definitions shown in this paper with the propositions as lemmas to a theorem prover, which is then used to derive further propositions or solve "object-level" tasks, application problems. Projection and circumscription can be processed by variants of second-order quantifier elimination for which a variety of techniques is available (see e.g. Gabbay et al. (2008); Wernhard (2009b)). Thus, a second approach would be the use a theorem prover that is extended by dedicated second-order quantifier elimination techniques. An approach for handling scopes which include only subsets of the ground atoms with a given predicate has been described in (Wernhard, 2004). The third approach is to resort to propositional techniques. Adaptions of DPLL SAT solving techniques are, for example, available for Boolean variable elimination (Huang and Darwiche, 2005; Wernhard, 2009b). In addition, recent SAT preprocessors involve variable elimination techniques (e.g. Heule et al. (2010)). For some tasks that involve projection or circumscription also QBF or SAT solvers can be applied.

A prototype system based on propositional logic has been implemented to explore the approach to computational processing of logics by eliminating operators such as `project`, `raise` and `circ` (Wernhard, 2011).[1] A macro feature allows the user to define additional operators like `gwsc` in terms of these. The core operation of the system is to take a formula with such operators and return an equivalent propositional formula where these operators are eliminated. Output formulas are simplified, and various functions for pretty printing them, for example to display their models, are provided. The system uses Prolog as an environment that allows to pass intermediate results through logic variables between its components. It includes features which facilitate the preparation of propositional encodings of applications by permitting compound terms as propositional atoms and providing support for schematic formula expansion. The system provides a uniform user interface that integrates a portfolio of embedded methods and external programs. By applying propositions presented in this paper, input formulas are rewritten such that suitable subproblems can be passed to external QBF or SAT solvers, or be handled by dedicated elimination procedures, which currently are implemented naively, adequate for small applications.

Acknowledgements

The author would like to thank the participants of FTP'09 for their stimulating questions and to the anonymous referees for helpful suggestions to improve the presentation.

Appendix. Proof of Proposition 27

We use the notation and symbols as specified for the main part of the paper in Section 2. In addition, we use the following abbreviations:

con. for *contracting the definition of*,
exp. for *expanding the definition of*.

[1] Available from http://cs.christophwernhard.com/toyelim/.

We show proofs of Propositions 27.i–27.iii. They are preceded by two auxiliary propositions: Proposition 28 is used to prove Proposition 29, which in turn is referenced in the proof of Proposition 27.ii.

Proposition 28. *If* $F \models G$ *then*

$$\mathsf{project}_S(F) \wedge \neg\mathsf{raise}_S(G) \models \mathsf{project}_{S\cup\overline{S}}(F).$$

Proof. Consider the table below. Assume (1), that is, the precondition of the proposition. Let $\langle I, \beta\rangle$ be an interpretation such that (2) holds.

(1)	$F \models G$.	assumption
(2)	$\langle I, \beta\rangle \models \mathsf{project}_S(F) \wedge \neg\mathsf{raise}_S(G)$.	assumption
(3)	$\mathsf{raise}_S(F) \models \mathsf{raise}_S(G)$.	by (1), Proposition 13.i
(4)	$\langle I, \beta\rangle \models (\mathsf{project}_{S\cup\overline{S}}(F) \vee \mathsf{raise}_S(F)) \wedge \neg\mathsf{raise}_S(G)$.	by (2), Proposition 13.viii
(5)	$\langle I, \beta\rangle \models \mathsf{project}_{S\cup\overline{S}}(F)$.	by (4),(3) □

Proposition 29. *If* $F \models G$ *then*

$$\mathsf{project}_S(F) \wedge \mathsf{circ}_S(G) \models \mathsf{project}_{S\cup\overline{S}}(F).$$

Proof. Follows from Proposition 28, since by the definition of circ it holds that $\mathsf{circ}_S(G) \models \neg\mathsf{raise}_S(G)$. □

Proposition 27.i. If F is well-founded with respect to S and $G \equiv \mathsf{project}_S(G)$, then

$$\mathsf{circ}_S(F) \models G \quad \textit{iff} \quad F \models G.$$

Proof. Assume the preconditions of the proposition:

(1)	F is well-founded with respect to S.	assumption
(2)	$G \equiv \mathsf{project}_S(G)$.	assumption

Consider the table below.

(3)	$\mathsf{circ}_S(F) \models G$	
(4)	iff $\mathsf{project}_S(\mathsf{circ}_S(F)) \models G$	by (2), Proposition 7.vi
(5)	iff $\mathsf{project}_S(F) \models G$	by (1), Proposition 21
(6)	iff $F \models G$.	by (2), Proposition 7.vi □

Proposition 27.ii. If F is well-founded with respect to S and $G \equiv \mathsf{project}_{S\cup\overline{S}}(G)$, then

$$\mathsf{circ}_S(F) \models G \quad \textit{iff} \quad F \models \mathsf{gsnc}_S(F, G).$$

Proof. Assume the preconditions of the proposition:

(1)	F is well-founded with respect to S.	assumption
(2)	$G \equiv \mathsf{project}_{S\cup\overline{S}}(G)$.	assumption

Left-to-right: Consider the table below. Assume (3), that is, the left side of the proposition.

(3)	$\mathsf{circ}_S(F) \models G$.	assumption
(4)	$\mathsf{circ}_S(F) \models F \wedge G$.	by (3), Proposition 16.i
(5)	$\mathsf{circ}_S(F) \models \mathsf{project}_S(F \wedge G)$.	by (4), Proposition 7.i
(6)	$F \models \mathsf{project}_S(F \wedge G)$.	by (5), (1), Proposition 27.i
(7)	$F \models \mathsf{gsnc}_S(F, G)$.	by con. gsnc

To derive step (6), the preconditions of Proposition 27.i are met since

$$\mathsf{project}_S(F \wedge G) \equiv \mathsf{project}_S(\mathsf{project}_S(F \wedge G)), \tag{x}$$

which follows from Proposition 7.v. Right-to-left: Consider the table below. Assume (8), that is, the right side of the proposition.

(8)	$F \models \mathsf{gsnc}_S(F, G)$.	assumption
(9)	$F \models \mathsf{project}_S(F \wedge G)$.	by exp. gsnc
(10)	$\mathsf{circ}_S(F) \models \mathsf{project}_S(F \wedge G)$.	by (9), Proposition 16.i
(11)	$\mathsf{circ}_S(F) \models \mathsf{project}_{S\cup\overline{S}}(F \wedge G)$.	by (10) and Proposition 29
(12)	$\mathsf{circ}_S(F) \models \mathsf{project}_{S\cup\overline{S}}(G)$.	by (11) and Proposition 7.xvii
(13)	$\mathsf{circ}_S(F) \models G$.	by (12) and (2)

To derive step (11), the left side of Proposition 29 is matched by $\mathsf{project}_S(F \wedge G) \wedge \mathsf{circ}_S(F)$, which by (10) is equivalent to $\mathsf{circ}_S(F)$. □

Proposition 27.iii. If F is well-founded with respect to S, then

$$\mathsf{circ}_S(F) \models G \quad \textit{iff} \quad F \models \mathsf{gsnc}_S(F, \mathsf{gwsc}_{\overline{S}}(F, G)).$$

Proof. Assume the precondition of the proposition:

(1)	F is well-founded with respect to S.	assumption

We are going to prove the following equivalences:

$$\mathsf{gwsc}_{\overline{S}}(F, G) \equiv \mathsf{project}_{S\cup\overline{S}}(\mathsf{gwsc}_{\overline{S}}(F, G)). \tag{xi}$$

$$\mathsf{circ}_S(F) \models G \quad \text{if and only if} \quad \mathsf{circ}_S(F) \models \mathsf{gwsc}_{\overline{S}}(F, G). \tag{xii}$$

The proposition then follows from these equivalences, the assumption of well-foundedness, and Proposition 27.ii, as shown in the following table:

(2)	$\mathsf{circ}_S(F) \models G$	
(3)	iff $\mathsf{circ}_S(F) \models \mathsf{gwsc}_{\overline{S}}(F, G)$	by (xii)
(4)	iff $F \models \mathsf{gsnc}_S(F, \mathsf{gwsc}_{\overline{S}}(F, G))$.	by (xi), (1), 27.ii

Equivalence (xi) can be shown as in the following table:

(5)	$\mathsf{gwsc}_{\overline{S}}(F, G)$	
(6)	$\equiv \neg\mathsf{project}_S(F \wedge \neg G)$	by exp. gwsc
(7)	$\equiv \mathsf{project}_{\overline{S}}(\neg\mathsf{project}_S(F \wedge \neg G))$	by Proposition 7.xxi
(8)	$\equiv \mathsf{project}_{S\cup\overline{S}}(\mathsf{project}_{\overline{S}}(\neg\mathsf{project}_S(F \wedge \neg G)))$	by Proposition 7.v
(9)	$\equiv \mathsf{project}_{S\cup\overline{S}}(\mathsf{gwsc}_{\overline{S}}(F, G))$.	by Proposition 7.xxi, con. gwsc

The left-to-right direction of equivalence (xii) can be shown as follows: Consider the table below. Assume (10), that is, the left side of equivalence (xii).

(10)	$\mathsf{circ}_S(F) \models G$.	assumption
(11)	$F \wedge \neg\mathsf{raise}_S(F) \models G$.	by (10), exp. circ
(12)	$F \wedge \neg G \models \mathsf{raise}_S(F)$.	by (11), logical equivalence
(13)	$\mathsf{project}_S(F \wedge \neg G) \models \mathsf{raise}_S(F)$.	by (12), Propositions 13.ix and 7.vi
(14)	$\neg\mathsf{raise}_S(F) \models \neg\mathsf{project}_S(F \wedge \neg G)$.	by (13), logical equivalence
(15)	$F \wedge \neg\mathsf{raise}_S(F) \models \neg\mathsf{project}_S(F \wedge \neg G)$.	by (14), logical entailment
(16)	$\mathsf{circ}_S(F) \models \mathsf{gwsc}_{\overline{S}}(F \wedge \neg G)$.	by (15), con. circ, con. gwsc

The right-to-left direction of equivalence (xii) can be shown as follows: Consider the table below. Assume (17), that is, the right side of equivalence (xii).

(17)	$\mathsf{circ}_S(F) \models \mathsf{gwsc}_{\overline{S}}(F, G)$.	assumption
(18)	$\mathsf{circ}_S(F) \models \neg\mathsf{project}_S(F \wedge \neg G)$.	by (17), exp. gwsc
(19)	$\mathsf{circ}_S(F) \models \neg(F \wedge \neg G)$.	by (18), Proposition 7.i
(20)	$\mathsf{circ}_S(F) \models G$.	by (19), Proposition 16.i □

References

Doherty, P., Łukaszewicz, W., Szałas, A., 1997. Computing circumscription revisited: a reduction algorithm. Journal of Automated Reasoning 18 (3), 297–338.

Doherty, P., Łukaszewicz, W., Szałas, A., 2001. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence. IJCAI-01. Morgan Kaufmann, pp. 145–151.

Ebbinghaus, H.-D., Flum, J., Thomas, W., 1984. Mathematical Logic. Springer, New York.
Gabbay, D.M., Schmidt, R.A., Szałas, A., 2008. Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications, London.
Gelfond, M., Przymusinska, H., Przymusinski, T., 1986. The extended closed world assumption and its relationship to parallel circumscription. In: Proceedings of the ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. PODS'86. ACM, pp. 133–139.
Heule, M., Järvisalo, M., Biere, A., 2010. Clause elimination procedures for CNF formulas. In: Logic for Programming, Artificial Intelligence and Reasoning: 17th International Conference. LPAR 17. In: LNCS, vol. 6397. Springer, pp. 357–371.
Huang, J., Darwiche, A., 2005. DPLL with a trace: From SAT to knowledge compilation. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. IJCAI-05. Morgan Kaufmann, pp. 156–162.
Hölldobler, S., Philipp, T., Wernhard, C., 2011. An abductive model for human reasoning (poster paper). In: Logical Formalizations of Commonsense Reasoning, Papers from the AAAI 2011 Spring Symposium. In: AAAI Spring Symposium Series Technical Reports, AAAI Press, pp. 135–138.
Lang, J., Liberatore, P., Marquis, P., 2003. Propositional independence — formula-variable independence and forgetting. Journal of Artificial Intelligence Research 18, 391–443.
Lifschitz, V., 1986. Pointwise circumscription: preliminary report. In: Proceedings of the Fifth National Conference on Artificial Intelligence, AAAI'86, vol. 1. Morgan Kaufmann, pp. 406–410.
Lifschitz, V., 1994. Circumscription. In: Gabbay, D.M., Hogger, C.J., Robinson, J.A. (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3. Oxford University Press, Oxford, pp. 298–352.
Lin, F., 2001. On strongest necessary and weakest sufficient conditions. Artificial Intelligence 128, 143–159.
McCarthy, J., 1980. Circumscription — a form of non-monotonic reasoning. Artificial Intelligence 13, 27–39.
Przymusinski, T., 1989. An algorithm to compute circumscription. Artificial Intelligence 83, 59–73.
Stenning, K., van Lambalgen, M., 2008. Human Reasoning and Cognitive Science. MIT Press, Cambridge, MA.
Wernhard, C., 2004. Semantic knowledge partitioning. In: Logics in Artificial Intelligence: 9th European Conference. JELIA 04. In: LNAI, vol. 3229. Springer, pp. 552–564.
Wernhard, C., 2008. Literal projection for first-order logic. In: Logics in Artificial Intelligence: 11th European Conference. JELIA 08. In: LNAI, vol. 5293. Springer, pp. 389–402.
Wernhard, C., 2009a. Automated Deduction for Projection Elimination. No. 324 in Dissertationen zur Künstlichen Intelligenz (DISKI). AKA/IOS Press, Heidelberg/Amsterdam.
Wernhard, C., 2009b. Tableaux for projection computation and knowledge compilation. In: Automated Reasoning with Analytic Tableaux and Related Methods: 18th International Conference, TABLEAUX 2009. In: LNAI, vol. 5607. Springer, pp. 325–340.
Wernhard, C., 2010a. Circumscription and projection as primitives of logic programming. In: Technical Communications of the 26th International Conference on Logic Programming. ICLP'10. In: Leibniz International Proceedings in Informatics (LIPIcs), vol. 7. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 202–211.
Wernhard, C., 2010b. Literal projection and circumscription. In: Proceedings of the 7th International Workshop on First-Order Theorem Proving, FTP'09. Vol. 556 of CEUR Workshop Proceedings. CEUR-WS.org, pp. 60–74.
Wernhard, C., 2011. Computing with logic as operator elimination: the ToyElim system. In: 25th Workshop on Logic Programming, WLP 2011, Infsys Research Report 1843-11-06, Technische Universität Wien, pp. 94–98.